Postel

Network Working Group
Request for Comments: 1085

M. Rose
TWG
December 1988

ISO Presentation Services on top of TCP/IP-based internets

Status of this Memo

This memo proposes a standard for the Internet community. Distribution of this memo is unlimited.

1. Introduction

[RFC1006] describes a mechanism for providing the ISO transport service on top of the Transmission Control Protocol (TCP) [RFC793] and Internet Protocol (IP) [RFC791]. Once this method is applied, one may implement "real" ISO applications on top of TCP/IP-based internets, by simply implementing OSI session, presentation, and application services on top of the transport service access point which is provided on top of the TCP. Although straight-forward, there are some environments in which the richness provided by the OSI application layer is desired, but it is nonetheless impractical to implement the underlying OSI infrastructure (i.e., the presentation, session, and transport services on top of the TCP). This memo describes an approach for providing "stream-lined" support of OSI application services on top of TCP/IP-based internets for such constrained environments.

2. Terminology

In as much as this memo is concerned primarily with concepts defined in the framework of Open Systems Interconnection (OSI) as promulgated by the International Organization for Standardization (ISO), the terminology used herein is intended to be entirely consistent within that domain of discourse. This perspective is being taken despite the expressed intent of implementing the mechanism proposed by this memo in the Internet and other TCP/IP-based internets. For those more familiar with the terminology used in this latter domain, the author is apologetic but unyielding.

Although no substitute for the "correct" definitions given in the appropriate ISO documents, here is a short summary of the terms used herein.

Application Context:
The collection of application service elements which cooperatively interact within an application-entity.

Application Service Element:
A standardized mechanism, defined by both a service and a protocol, which provides a well-defined capability, e.g.,

ROSE - the Remote Operations Service Element, which orchestrates the invocation of "total" operations between application-entities [ISO9066/2].

ACSE - the Association Control Service Element, which manages associations between application entities [ISO8650].

Object Identifier:
An ordered set of integers, used for authoritative identification.

Presentation Service:
A set of facilities used to manage a connection between two application-entities. The fundamental responsibility of the presentation service is to maintain transfer syntaxes which are used to serialize application protocol data units for transmission on the network and subsequent de-serialization for reception.

Protocol Data Unit (PDU):
A data object exchanged between service providers.

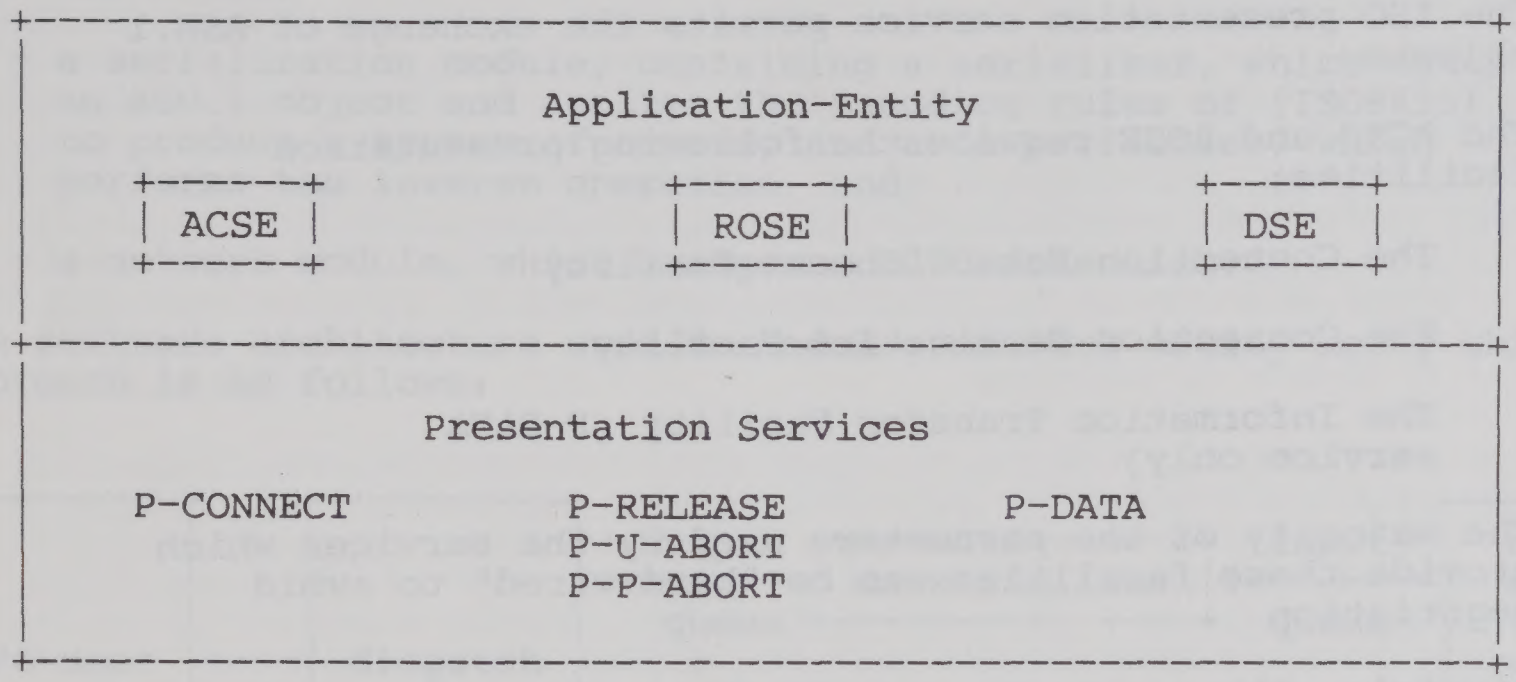
Serialization:
The process of applying an abstract transfer notation to an object described using abstract syntax notation one (ASN.1) [ISO8824] in order to produce a stream of octets. De-serialization is the inverse process.

It is assumed that the reader is familiar with terminology pertaining to the reference model [ISO7498], to the service conventions in the model [ISO8509], and to the connection-oriented presentation service [ISO8822].

3. Scope

The mechanism proposed by this memo is targeted for a particular class of OSI applications, namely those entities whose application context contains only an Association Control Service Element (ACSE) and a Remote Operations Service Element (ROSE). In addition, a

Directory Services Element (DSE) is assumed for use by the application-entity, but only in a very limited sense. The organization of such an entity is as follows:



The mechanism proposed by this memo is not applicable to entities whose application context is more extensive (e.g., contains a Reliable Transfer Service Element). The mechanism proposed by this memo could be modified to support additional elements. However, such extensions would, at this time, merely serve to defeat the purpose of providing the minimal software infrastructure required to run the majority of OSI applications.

The motivation for this memo was initially derived from a requirement to run the ISO Common Management Information Protocol (CMIP) in TCP/IP-based internets. In its current definition, CMIP uses precisely the application service elements provided for herein. It may be desirable to offer CMIP users a quality of service different than the one offered by a connection with a high-quality level of reliability. This would permit a reduced utilization of connection-related resources. This memo proposes a mechanism to implement this less robust -- and less costly -- quality of service.

4. Approach

The approach proposed by this memo relies on the following architectural nuances:

- the TCP is a stream-oriented transport protocol

- ASN.1 objects, when represented as a stream of octets are self-delimiting

- The ISO presentation service permits the exchange of ASN.1 objects

- The ACSE and ROSE require the following presentation facilities:

    The Connection Establishment Facility

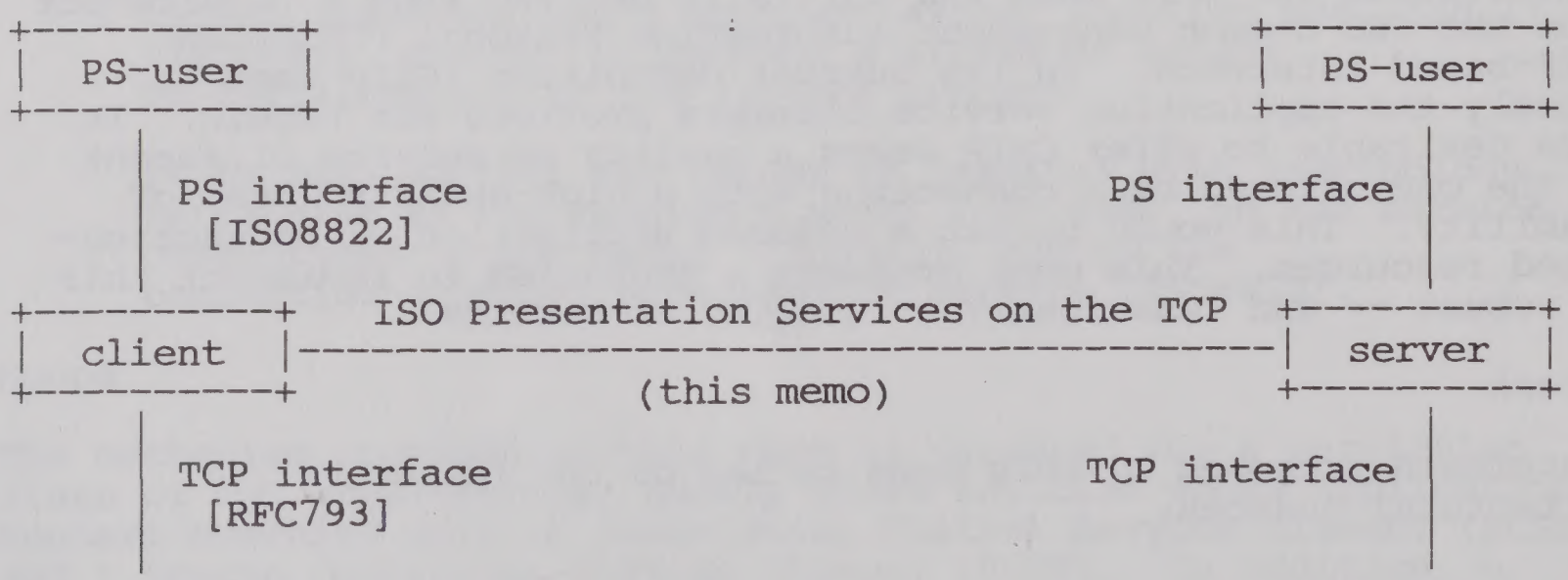
    The Connection Termination Facility

    The Information Transfer Facility (P-DATA service only)

- The majority of the parameters used by the services which provide these facilities can be "hard-wired" to avoid negotiation

In principle, these nuances suggest that a "cheap" emulation of the ISO presentation services could be implemented by simply serializing ASN.1 objects over a TCP connection. This approach is precisely what is proposed by this memo.

Given this perspective, this memo details how the essential features of the ISO presentation service may be maintained while using a protocol entirely different from the one given in [ISO8823]. The overall composition proposed by this memo is as follows:



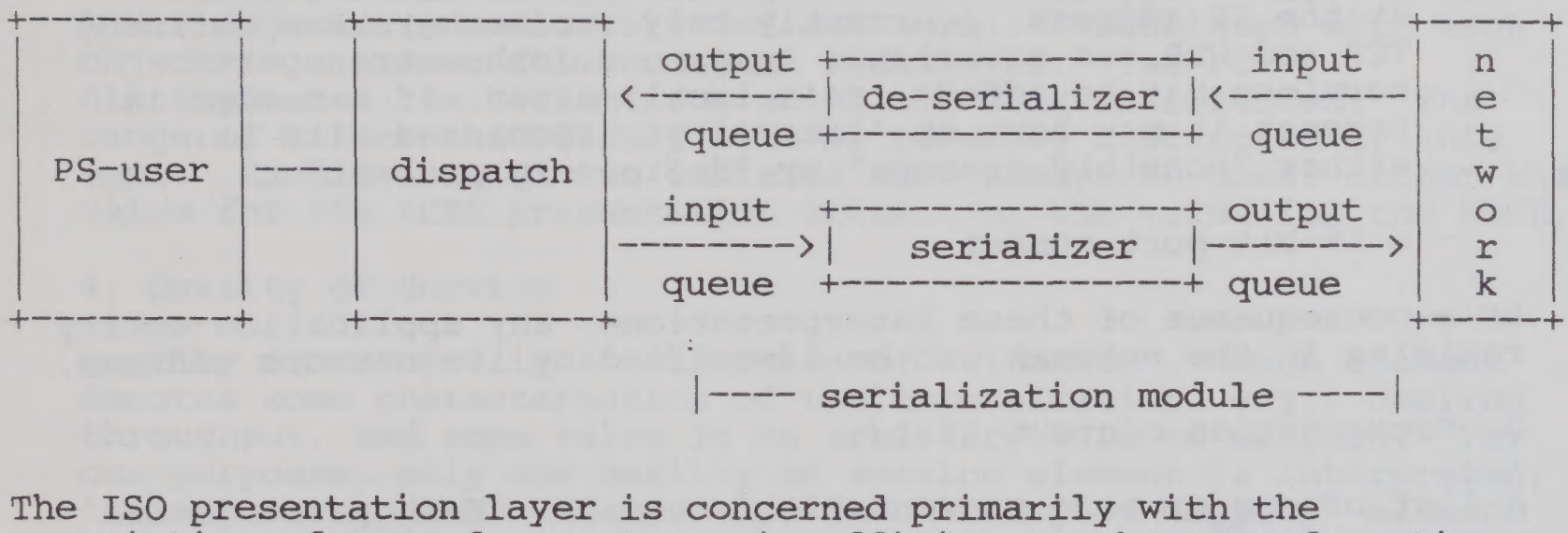
In greater detail, the "client" and "server" boxes implement the protocol described in this memo. Each box contains three modules:

- a dispatch module, which provides the presentation services interface,

- a serialization module, containing a serializer, which takes an ASN.1 object and applies the encoding rules of [ISO8825] to produce a stream of octets, and a de-serializer, which performs the inverse operation, and

- a network module, which manages a TCP connection.

The software architecture used to model a network entity using this approach is as follows:



The ISO presentation layer is concerned primarily with the negotiation of transfer syntaxes in addition to the transformation to and from transfer syntax. However, using the mechanism proposed by this memo, no negotiation component will be employed. This memo specifies the fixed contexts which exist over each presentation connection offered. This memo further specifies other constants which are used in order to eliminate the need for presentation layer negotiation.

5. Fundamental Parameters

There are certain parameters which are used by the presentation service and are defined here.

1. Presentation address:

The structure of a presentation address is presented in Addendum 3 to [ISO7498]. This memo interprets a presentation address as an

ordered-tuple containing:

- one or more network addresses
- a transport selector
- a session selector
- a presentation selector

Each selector is an uninterpreted octet string of possibly zero length. The mechanism proposed in this memo completely ignores the values of these selectors. Note however that the value of the presentation selector is preserved by the provider.

A network address is interpreted as containing three components:

- a 32-bit IP address

- a set indicating which transport services are available at the IP address (currently only two members are defined: TCP and UDP; as experience is gained, other transport services may be added); as a local matter, if a member is present it may have an "intensity" associated with it: either "possibly present" or "definitely present"

- a 16-bit port number

As a consequence of these interpretations, any application-entity residing in the network can be identified by its network address.

2. Presentation context list

A list of one or more presentation contexts. Each presentation context has three components:

- a presentation context identifier (PCI), an integer

- an abstract syntax name, an object identifier

- an abstract transfer name, an object identifier

The range of values these components may take is severely restricted by this memo. In particular, exactly two contexts are defined: one for association control and the other for the specific application service element which is being carried as ROS APDUs (see the section on connection establishment for the precise values).

In addition, if the presentation context list appears in a "result" list (e.g., the Presentation context result list

parameter for the P-CONNECT service), a fourth component is present:

- an acceptance indicator

which indicates if the context was accepted by both the service provider and the remote peer. If the context was not accept, a brief reason, such as "abstract syntax not supported" is given.

For the novice reader, one might think of the abstract syntax notation as defining the vocabulary of some language, that is, it lists the words which can be spoken. In contrast, the abstract transfer notation defines the pronunciation of the language.

3. User data

User data passes through the presentation service interface as ASN.1 objects (in a locally defined form). Associated with each object is a presentation context identifier. The PCI distinguishes the context for which the data is intended. The range of values the PCI may take is severely restricted by this memo. Exactly one of two contexts must always be used: either the value for the ACSE presentation context or the value for the ROSE.

4. Quality of Service

Quality of service is a collection of "elements". Each element denotes some characteristics of the communication, e.g., desired throughput, and some value in an arbitrary unit of measure. For our purposes, only one quality of service element is interpreted, "transport-mapping". Currently, the "transport-mapping" element takes on one of two values: "tcp-based" or "udp-based". At present, the two values may also be referred to as "high-quality" or "low-quality", respectively.

As experience is gained, other values may be added. These values would correspond directly to the new transport services which are listed in the network address.

5. Version of Session Service

Some application service elements (e.g., the ACSE) invoke different procedures based on the (negotiated) version of the session service available. Implementations of this memo always indicate that session service version 2 has been negotiated.

6. Choice of Transport Service

Discussion thus far has centered along the use of the TCP as the underlying transport protocol. However, it has also been noted that it may be desirable to permit a quality of service with less reliability in order to take advantage of some other characteristic of the transport service.

The introduction of this service has several profound impacts on the model, and it is beyond the scope of this memo to enumerate these impacts. However, this memo does propose a mechanism by which such a facility is implemented.

To begin, we use the quality of service parameter for the P-CONNECT service to select an underlying transport service. Only one element is currently interpreted, "transport-mapping" which takes the value "tcp-based" or "udp-based". If the value is "tcp-based", then the presentation provider will use TCP as the underlying transport service. If, however, the value of "transport-mapping" is "udp-based", then the presentation provider will use the UDP instead.

The User Datagram Protocol (UDP) [RFC768] is used to implement the udp-based service. Very few transport-level facilities are placed on top of the UDP service, i.e., it is not the intent of this memo to "re-invent" the facilities in the TCP. Hence, It is critical to understand that

low-quality means LOW-QUALITY!

Because the UDP is a packet-oriented protocol, it is necessary to slightly redefine the role of the serialization module. For the serializer, we say that each top-level ASN.1 object placed on the input queue will form a single UDP datagram on the output queue which is given to the network. Similarly, for the de-serializer, we say that each UDP datagram placed on the input queue from the network will form a single top-level ASN.1 object placed on the output queue. The term "top-level ASN.1 object" refers, of course, to the protocol data units being exchanged by the presentation providers.

It should be noted that in its current incarnation, this memo permits the choice of two different transport protocols, e.g., the TCP or the UDP. However, as experience is gained and as other transport protocols are deployed (e.g., the VMTP), then future incarnations of this memo will permit these transport protocols to be used. This is a three step process: first, the set of transport services defined for the network address is updated; second, a corresponding value is added to the range of the quality of service element "transport-mapping"; and, third, the following sections of this memo are

modified accordingly.

7. Connection Establishment

The Connection Establishment facility consists of one service, the P-CONNECT service.

7.1. The P-CONNECT Service

This service is used to bring two identified application-entities into communication. Its successful use results in a presentation connection, with an initial defined context set, being established between then. This connection is available for their subsequent communication. This is a confirmed service whose effects are sequenced and non-destructive.

If the udp-based service is selected, then a presentation connection is formed which should be used infrequently and will have minimal reliability characteristics.

For our purposes, the P-CONNECT service:

- requests TCP or UDP resources,
- builds a fixed defined context set, and
- exchanges initial user data.

Following are the interpretation of and the defaults assigned to the parameters of the P-CONNECT service:

1. Calling Presentation Address

   This is a presentation address. Although the ISO presentation service states that this parameter is mandatory, in practice, a local implementation rule may be used to determine an "ephemeral" address to use.

2. Called Presentation Address

   This is a presentation address. Note that when issuing the P-CONNECT.REQUEST primitive, this parameter may contain more than one network address. In the P-CONNECT.INDICATION primitive however, only one network address, the one actually used to establish the presentation connection, is present. (Appendix C describes a strategy which might be used to determine the actual network address).

3. Responding Presentation Address

This parameter is identical to the value of the Called Presentation Address parameter of the P-CONNECT.INDICATION primitive.

4. Multiple defined Contexts

Always TRUE. Note that this parameter is present only in the DIS version of the presentation service.

5. Presentation context definition list

Two contexts are defined:

PCI	Abstract Syntax Name	Abstract Transfer Name
1	specific to the application	"iso asn.1 abstract transfer" 1.0.8825
3	"acse pci version 1" 2.2.1.0.0	"iso asn.1 abstract transfer" 1.0.8825

The abstract syntax and transfer names for the ACSE PCI are for use with the DIS version of association control. If the IS version is being used, then this PCI is used instead:

3	"acse pci version 1" 2.2.1.0.1	"asn.1 basic encoding" 2.1.1

6. Presentation context result list

Identical to the Presentation context definition list with the addition that the acceptance indicator for both contexts is "accepted".

7. Default Context Name

None.

8. Default Context Result

Not applicable.

9. Quality of Service

The element "transport-mapping" takes the value "tcp-based" or "udp-based". In the future the range of values may be extended.

10. Presentation Requirements

None (the kernel functional unit is always used).

11. Session Requirements

Full duplex.

12. Initial synchronization point serial number

None.

13. Initial Assignment of tokens

None.

14. Session connection identifier

Unlike the "real" presentation service, depending on the quality of service selected, this parameter may have great significance to presentation provider. Hence, the following format of the session connection identifier is mandated by this memo.

user data: a local string encoded as a T.61 string using ASN.1, e.g., given string "gonzo":

```
14     05     67   6f   6e   7a   6f
tag  length   "g"  "o"  "n"  "z"  "o"
```

common data: a universal time encoding using ASN.1, e.g., given time "880109170845":

```
17     0c     38   38   30   31   30   ...
tag  length   "8"  "8"  "0"  "1"  "0"  ...
```

additional data: any string encoded as a T.61 string using ASN.1 (optional)

As a local convention, the presentation provider may disregard the first two octets of each data component for transmission on the network as when the session connection identifier is represented with ASN.1, the tag and length octets will be added anyway.

15. User Data

A single ASN.1 object is present, the appropriate A-ASSOCIATE PDU, carried in presentation context 3.

16. Result

One of the following values: acceptance, user-rejection, provider-rejection (transient), or provider-rejection (permanent).

8. Connection Termination

The Connection Termination facility consists of three services, the P-RELEASE, P-U-ABORT, and P-P-ABORT services.

8.1. The P-RELEASE Service

This service provides the service user with access to a negotiated release facility. This service has effects which are sequenced and non-destructive. Either presentation user is permitted to request this service. However, in the event of collision, a provider-initiated abort procedure will be invoked.

If the udp-based service is selected, then any data in transit may be discarded.

For our purposes, the P-RELEASE service:

- waits for the serialization module to drain,

- sends release user data, and

- releases TCP or UDP resources

Following are the interpretation of and the defaults assigned to the parameters of the P-RELEASE service:

1. Result

Release accepted.

2. User data

A single ASN.1 object is present, the appropriate A-RELEASE PDU,

8.2. The P-U-ABORT Service

This service can be used by either presentation user to force the release of a presentation connection at any time and have the correspondent presentation user informed of this termination. This service has effects which are not sequenced with respect to preceding service invocations and may be destructive. It does not require the agreement of both service users.

For our purposes, the P-U-ABORT service:

- flushes the serialization module,

- sends abort user data, and

- releases TCP or UDP resources

Following are the interpretation of and the defaults assigned to the parameters of the P-U-ABORT service:

1. Presentation context identifier list

   Contained in the ASN.1 objects, if any, that are delivered as user data.

2. User data

   A single ASN.1 object is present, an A-ABORT PDU, carried in presentation context 3.

8.3. The P-P-ABORT Service

This service is the means by which the service provider may indicate the termination of the presentation connection for reasons internal to the service provider. This service has effects which are not sequenced with respect to preceding service invocations. The execution of this service disrupts any other concurrently active service and may thus be destructive.

For our purposes, the P-P-ABORT service:

- flushes the serialization module, and

- releases TCP or UDP resources

Following are the interpretation of and the defaults assigned to the parameters of the P-P-ABORT service.

1. Provider reason

An integer code detailing why the connection was aborted. Codes include, but are not limited to: invalid PPDU parameter, unexpected PPDU, unrecognized PPDU, and specified reason.

2. Abort data

None.

9. Information Transfer

Although the Information Transfer facility consists of many services, only one, the P-DATA service, is provided by this memo.

9.1. The P-DATA Service

This services provides the service user with a data transfer capability. This service has effects which are sequenced and non-destructive.

If the udp-based service is selected, then there is an upper-bound on the size of the serialized ASN.1 objects which may be transmitted. This limit, imposed by the UDP, is 65536 octets. As a practical matter, it is probably a good idea to keep datagrams less than or equal to 536 octets in size.

For our purposes, the P-DATA service:

- sends user data

Following are the interpretation of and the defaults assigned to the parameters of the P-DATA service:

1. User data

A single ASN.1 object is present, a remote operations APDU, carried in presentation context 1.

10. Elements of Procedure

The service provider is in one of the following states:

IDLE, WAIT1, WAIT2, DATA, WAIT3, or WAIT4

The possible events are:

PS-user P-CONNECT.REQUEST

```
                        P-CONNECT.RESPONSE
                        P-RELEASE.REQUEST
                        P-RELEASE.RESPONSE
                        P-DATA.REQUEST
                        P-U-ABORT.REQUEST

      network           TCP closed or errored(*)
                        receive ConnectRequest PDU
                        receive ConnectResponse PDU
                        receive ReleaseRequest PDU
                        receive ReleaseResponse PDU
                        receive UserData(*) or CL-UserData(**) PDU
                        receive user-initiated Abort PDU
                        receive provider-initiated Abort PDU
                        timer expires(**)
```

The possible actions are:

```
   PS-user              P-CONNECT.INDICATION
                        P-CONNECT.CONFIRMATION
                        P-RELEASE.INDICATION
                        P-RELEASE.CONFIRMATION
                        P-DATA.INDICATION
                        P-U-ABORT.INDICATION
                        P-P-ABORT.INDICATION

   network              open TCP(*)
                        close TCP(*)
                        send ConnectRequest PDU
                        send ConnectResponse PDU
                        send ReleaseRequest PDU
                        send ReleaseResponse PDU
                        send UserData(*) or CL-UserData(**) PDU
                        send user-initiated Abort PDU
                        send provider-initiated Abort PDU
                        set timer(**)

      (*)   tcp-based service only
      (**)  udp-based service only
```

10.1. Elements of Procedure specific to the tcp-based service

The provider maintains the following information for each presentation connection:

- a local designator for the PS-user

- a local designator for a TCP connection

- the state of the connection (e.g., IDLE, WAIT1, and so on)

Upon receiving an event from the network, the provider finds the associated presentation connection. Matching is done by simply comparing local designators for the TCP connection. Whenever a connection remains in or returns to the IDLE state, any associated resources, such as an attachment to a local TCP port, are released.

In the procedures which follow, outgoing PDUs are "placed on the input queue for the serializer". This has a different meaning depending on the type of PDU being enqueued. If the PDU is not an abort PDU (user-initiated or provider-initiated), then the PDU is simply appended to the input queue regardless of the number of PDUs present. If however, the PDU is an abort PDU, then the provider checks the size of the input queue. If the input queue is non-empty or if the serializer is busy transmitting to the network, then the abort PDU is discarded, and the serializer is flushed, aborting any output to the network in progress. However, if the input queue is empty, then the Abort PDU is appended to the queue, and a small timer started. If the timer expires before the PDU has been serialized and transmitted, then the serializer is flushed, aborting any output to the network in progress.

Further, in general, whenever the TCP connection is closed (either locally by the provider, or remotely by the network) or has errored, the serializer is flushed. The one exception to this is if a ReleaseResponse PDU is being serialized and transmitted to the network. In this case, the provider will not close the TCP connection until after the serializer has finished.

10.2. Elements of Procedure specific to the udp-based service

The provider maintains the following information for each presentation connection:

- a local designator for the PS-user

- the 32-bit IP address and 16-bit UDP port number of the initiating host

- the 32-bit IP address and 16-bit UDP port number of the responding host

- the session connection identifier used to establish the presentation connection

- a local designator for an UDP endpoint

- the state of the connection (e.g., IDLE, WAIT1, and so on)

- a retransmission counter

Upon receiving an event from the network, the provider finds the associated presentation connection. Matching is done on the basis of addresses, ports, and the session connection identifier (i.e., two different presentation connections may differ only in their session connection identifier). If no presentation connection can be found, then for the purposes of discussion, it may be assumed that a "vanilla" presentation connection is created and initialized to the IDLE state. Further, whenever a connection remains in or returns to the IDLE state, any associated resources, such as an attachment to a local UDP port, are released.

In the procedures which follow, outgoing PDUs are "placed on the input queue for the serializer". This means that the ASN.1 object is serialized and the resulting sequence of octets is sent as a single UDP datagram.

10.3. State Transitions

Following are the rules for transitioning states. If an event associated with a user-generated primitive is omitted, then it is an interface error for the user to issue that primitive in the given state. Each state considers all possible incoming PDUs.

We assume that for the tcp-based service, that some entity starts a passive TCP open. When the passive open completes, the entity, using some local rule, locates a PS-user to be associated with the incoming presentation connection. This presentation connection is then placed in the IDLE state. The entity then continues listening for other passive opens to complete. The mechanisms associated with this entity are entirely a local matter, the concept of this listener is introduced solely as a modeling artifact.

Finally, if the udp-based service is selected, then CL-UserData PDUs are exchanged by the provider instead of UserData PDUs.

IDLE state

Event: P-CONNECT.REQUEST primitive issued

Based on the quality of service parameter and the list of network addresses in the called presentation address parameter, the provider

selects an address for the use of the presentation connection. The method for making this determination is a local matter. (Appendix C discusses a strategy which might be used.) For the discussion that follows, we assume that a network address supporting the desired quality of service has been determined.

Based on the network address chosen from the called presentation address parameter, the provider selects a compatible network address from the calling presentation address parameter. The provider attaches itself to the port associated with this network address. (By local determination, this address need not be used, and an "ephemeral" port may be chosen by the provider.)

For the tcp-based service, the provider attempts to establish a TCP connection to the network address listed in the called presentation address. If the connection can not be established, the P-CONNECT.CONFIRMATION(-) primitive is issued with a reason of provider-rejection, and the provider remains in the IDLE state.

Regardless, the user data parameter is placed in a ConnectRequest PDU, which is put on the input queue for the serializer.

For the udp-based service, the provider sets the retransmission counter to a small value (e.g., 2), and now starts a small timer.

Regardless, the provider enters the WAIT1 state.

Event: ConnectRequest PDU received

The provider issues the P-CONNECT.INDICATION primitive and enters the WAIT2 state.

Event: any other PDU received

If the PDU is not an Abort PDU, the provider constructs a provider-initiated Abort PDU, which is put on the input queue for the serializer. Regardless, the provider remains in the IDLE state.

WAIT1 state

Event: P-U-ABORT.REQUEST primitive issued

The user data parameter is placed in an Abort PDU, which is put on the input queue for the serializer. The provider enters the IDLE state.

Event: ConnectResponse PDU received

For the udp-based service, the timer is cancelled. If the PDU indicates rejection, the P-CONNECT.CONFIRMATION(-) primitive is issued and the provider enters the IDLE state. Otherwise, the P-CONNECT.CONFIRMATION(+) primitive is issued and the provider enters the DATA state.

Event: user-initiated Abort PDU received

The provider issues the P-U-ABORT.INDICATION primitive and enters the IDLE state.

Event: any other PDU received

If the PDU not an Abort PDU, the provider constructs a provider-initiated Abort PDU, which is put on the input queue for the serializer. Regardless, The provider issues the P-P-ABORT.INDICATION primitive and enters the the IDLE state.

Event: timer expires

The provider decrements the retransmission counter. If the resulting value is less than or equal to zero, the provider issues the P-CONNECT.CONFIRMATION(-) primitive and enters the IDLE state. Otherwise, a ConnectRequest PDU is put on the input queue for the serializer, the small timer is started again, and the provider remains in the WAIT1 state.

WAIT2 state

Event: P-CONNECT.RESPONSE primitive issued

The user data parameter is placed in a ConnectResponse PDU, which is put on the input queue for the serializer. If the result parameter had the value user-rejection, the provider enters the IDLE state. Otherwise if the parameter had the value acceptance, the provider enters the DATA state.

Event: P-U-ABORT.REQUEST primitive issued

The user data parameter is placed in an Abort PDU, which is put on the input queue for the serializer. The provider enters the IDLE state.

Event: user-initiated Abort PDU received

The provider issues the P-U-ABORT.INDICATION primitive and enters the IDLE state.

Event: any other PDU received

If the PDU is not an Abort PDU, the provider constructs a provider-initiated Abort PDU, which is put on the input queue for the serializer. Regardless, The provider issues the P-P-ABORT.INDICATION primitive and enters the the IDLE state.

DATA state

Event: P-DATA.REQUEST primitive issued

The user data parameter is placed in a UserData PDU, which is put on the input queue for the serializer. The provider remains in the DATA state.

Event: P-RELEASE.REQUEST primitive issued

The user data parameter is placed in a ReleaseRequest PDU, which is put on the input queue for the serializer.

For the udp-based service, the provider sets the retransmission counter to a small value (e.g., 2), and now starts a small timer.

Regardless, the provider enters the WAIT3 state.

Event: P-U-ABORT.REQUEST primitive issued

The user data parameter is placed in an Abort PDU, which is put on the input queue for the serializer. The provider enters the IDLE state.

Event: UserData PDU received

The provider issues the P-DATA.INDICATION primitive and remains in the DATA state.

Event: ReleaseRequest PDU received

The provider issues the P-RELEASE.INDICATION primitive, and enters the WAIT4 state.

Event: user-initiated Abort PDU received

The provider issues the P-U-ABORT.INDICATION primitive and enters the IDLE state.

Event: any other PDU received

If the PDU is not an Abort PDU, the provider constructs a provider-initiated Abort PDU, which is put on the input queue for the serializer. Regardless, the provider issues the P-P-ABORT.INDICATION primitive and enters the the IDLE state.

WAIT3 state

Event: P-U-ABORT.REQUEST primitive issued

The user data parameter is placed in an Abort PDU, which is put on the input queue for the serializer. The provider enters the IDLE state.

Event: ReleaseResponse PDU received

For the udp-based service, the timer is cancelled. The provider issues the P-RELEASE.CONFIRMATION primitive and enters the IDLE state.

Event: user-initiated Abort PDU received

The provider issues the P-U-ABORT.INDICATION primitive and enters the IDLE state.

Event: any other PDU received

If the PDU is not an Abort PDU, the provider constructs a provider-initiated Abort PDU, which is put on the input queue for the serializer. Regardless, the provider issues the P-P-ABORT.INDICATION primitive and enters the the IDLE state.

Event: timer expires

The provider decrements the retransmission counter. If the resulting value is less than or equal to zero, the provider constructs a provider-initiated Abort PDU, which is put on the input queue for the serializer. It then issues the P-P-ABORT.INDICATION primitive and enters the IDLE state. Otherwise, a ReleaseRequest PDU is put on the input queue for the serializer, the small timer is started again, and the provider remains in the WAIT3 state.

WAIT4 state

Event: P-RELEASE.RESPONSE primitive issued

The user data parameter is placed in a ReleaseResponse PDU, which is put on the input queue for the serializer. The provider now enters the IDLE state.

Event: P-U-ABORT.REQUEST primitive issued

The user data parameter is placed in an Abort PDU, which is put on the input queue for the serializer. The provider now enters the IDLE state.

Event: user-initiated Abort PDU received

The provider issues the P-U-ABORT.INDICATION primitive and enters the IDLE state.

Event: any other PDU received

If the PDU is not an Abort PDU, the provider constructs a provider-initiated Abort PDU, which is put on the input queue for the serializer. Regardless, the provider issues the P-P-ABORT.INDICATION primitive and enters the the IDLE state.

11. Directory Services

Although not properly part of the presentation service, this memo assumes and specifies a minimal Directory service capability for use by the application-entity.

The function of the Directory Service Element is to provide two mappings: first, a service name is mapped into an application entity title, which is a global handle on the service; and, second, the application-entity title is mapped onto a presentation address.

The structure of presentation addresses were defined in Section 5.

The structure of application-entity titles is less solidly agreed upon at the present time. Since objects of this type are not interpreted by the presentation service, this memo does not specify their structure. If the DIS version of association control is being used, then use of an OBJECT IDENTIFIER will suffice. If the IS version is being employed, then application-entity titles consist of two parts: an application-process title and an application-entity qualifier. It is suggested that the AP-Title use an OBJECT IDENTIFIER and that the AE-Qualifier use NULL.

This memo requires the following mapping rules:

1. The service name for an OSI application-entity using the mechanisms proposed by this memo is:

   <designator> "-" <qualifier>

   where <designator> is a string denoting either domain name or a 32-bit IP address, and <qualifier> is a string denoting the type of application-entity desired, e.g.,

   "gonzo.twg.com-mgmtinfobase"

2. Any locally defined mapping rules may be used to map the service designation into an application-entity title.

3. The application-entity title is then mapped into a presentation address, with uninterpreted transport, session, and presentation selectors, and one or more network addresses, each containing:

   -the 32-bit IP address resolved from the <designator> portion of the service name,

   - a set indicating which transport services are available

at the IP address,

- the 16-bit port number resolved from the <qualifier> portion of the service name (using the Assigned Numbers document), and

- optionally, a presentation selector, which is an uninterpreted sequence of octets.

The method by which the mappings are obtained are straight-forward. The directory services element employs the Domain Name System along with a local table which may be used to resolve the address employing local rules.

In the simplest of implementations, the DNS is used to map the <designator> to an IP address, and to fill-in the set of transport services available at the IP address. The port number is found in a local table derived from the current Assigned Numbers document. Finally, the presentation selector is empty.

A more ambitious implementation would use a local table to perhaps provide a presentation selector. This would be useful, e.g., in "proxy" connections. The network address would resolve to the proxy agent for the non-IP device, and the presentation selector would indicate to the proxy agent the particular non-IP device desired. This implies, of course, that the local table and the proxy agent bilaterally agree as to the interpretation of each presentation selector.

12. Remarks

To begin, if one really wanted to implement ISO applications in a TCP/IP-based network, then the method proposed by [RFC1006] is the preferred method for achieving this. However, in a constrained environment, where it is necessary to host an application layer entity with a minimal amount of underlying OSI infrastructure, this memo proposes an alternative mechanism. It should be noted that an OSI application realized using this approach can be moved directly to an [RFC1006]-based environment with no modifications.

A key motivation therefore is to minimize the size of the alternate underling infrastructure specified by this memo. As more and more presentation services functionality is added, the method proposed herein would begin to approximate the ISO presentation protocol. Since this in contrary to the key motivation, featurism must be avoided at all costs.

13. Acknowledgements

Several individuals contributed to the technical quality of this memo:

Karl Auerbach, Epilogue Technologies
Joseph Bannister, Unisys
Amatzia Ben-Artzi, Sytek
Stephen Dunford, Unisys
Lee Labarre, MITRE
Keith McCloghrie, The Wollongong Group
Jim Robertson, Bridge Communications
Glenn Trewitt, Stanford University

14. References

[ISO7498] Information Processing Systems - Open Systems Interconnection, "Basic Reference Model", October, 1984.

[ISO8509] Information Processing Systems - Open Systems Interconnection, " Service Conventions".

[ISO8650] Information Processing Systems - Open Systems Interconnection, " Protocol Specification for the Association Control Service Element (Final Text of DIS 8650)", January, 1988.

[ISO8822] Information Processing Systems - Open Systems Interconnection, " Connection Oriented Presentation Service Definition (Final Text of DIS 8822)", April, 1988.

[ISO8823] Information Processing Systems - Open Systems Interconnection, " Connection Oriented Presentation Protocol Specification (Final Text of DIS 8822)", April, 1988.

[ISO8824] Information Processing Systems - Open Systems Interconnection, " Specification of Abstract Syntax Notation One (ASN.1)", December, 1987.

[ISO8825] Information Processing Systems - Open Systems Interconnection, "Specification of basic encoding rules for Abstract Syntax Notation One (ASN.1)", December, 1987.

[ISO9072/2] Information Processing Systems - Text Communication MOTIS, " Remote Operations Part 2: Protocol

Specification (Working Document for DIS 9072/2)", November, 1987.

[RFC768] Postel, J., "User Datagram Protocol", RFC 768, USC/ISI, 28 August 1980.

[RFC791] Postel, J., "Internet Protocol - DARPA Internet Program Protocol Specification", RFC 791, USC/ISI, September 1981.

[RFC793] Postel, J., "Transmission Control Protocol - DARPA Internet Program Protocol Specification", RFC 793, USC/ISI, September 1981.

[RFC1006] Rose, M., and D. Cass, "ISO Transport 1 on Top of the TCP Version: 3", Northrop Research and Technology Center, May 1987.

Appendix A:

Abstract Syntax Definitions

```
RFC1085-PS DEFINITIONS ::=

BEGIN

PDUs ::=
        CHOICE {
connectRequest
    ConnectRequest-PDU,

connectResponse
    ConnectResponse-PDU,

releaseRequest
    ReleaseRequest-PDU,

releaseResponse
    ReleaseResponse-PDU,

abort
    Abort-PDU,

userData
    UserData-PDU,

cL-userData
    CL-UserData-PDU
```

```
    }

-- connect request PDU

ConnectRequest-PDU ::=
    [0]
        IMPLICIT SEQUENCE {
            version[0]              -- version-1 corresponds to to this
                                       memo
                IMPLICIT INTEGER { version-1(0) },

            reference
                SessionConnectionIdentifier,

            calling
                PresentationSelector
                OPTIONAL,

            called[2]
                IMPLICIT PresentationSelector
                OPTIONAL,

            asn[3]                  -- the ASN for PCI #1
                IMPLICIT OBJECT IDENTIFIER,

            user-data
                UserData-PDU
        }

SessionConnectionIdentifier ::=
    [0]
        SEQUENCE {
            callingSSUserReference
                T61String,

            commonReference
                UTCTime,

            additionalReferenceInformation[0]
                IMPLICIT T61String
                OPTIONAL
        }

PresentationSelector ::=
    [1]
        IMPLICIT OCTET STRING
```

```
-- connect response PDU

ConnectResponse-PDU ::=
    [1]
        IMPLICIT SEQUENCE {
            reference               -- present only in the udp-based
                                    -- service
                SessionConnectionIdentifier
                OPTIONAL,

            responding
                PresentationSelector
                OPTIONAL,

            reason[2]               -- present only if the connection
                                    -- was rejected
                IMPLICIT Rejection-reason
                OPTIONAL,

            user-data               -- present only if reason is absent
                                    -- OR has the
                                    -- value rejected-by-responder
                UserData-PDU
                OPTIONAL
        }

Rejection-reason ::=
        INTEGER {
            rejected-by-responder(0)
            called-presentation-address-unknown(1),
            local-limit-exceeded(3),
            protocol-version-not-supported(4),
        }


-- release request PDU

ReleaseRequest-PDU ::=
    [2]
        IMPLICIT SEQUENCE {
            reference               -- present only in the udp-based
                                    -- service
                SessionConnectionIdentifier
                OPTIONAL,

            user-data
                UserData-PDU
        }
```

```
-- release response PDU

ReleaseResponse-PDU ::=
    [3]
        IMPLICIT SEQUENCE {
            reference               -- present only in the udp-based
                                    -- service
                SessionConnectionIdentifier
                OPTIONAL,

            user-data
                UserData-PDU
        }

-- abort PDU

Abort-PDU ::=
    [4]
        SEQUENCE {
            reference               -- present only in the udp-based
                                    -- service
                SessionConnectionIdentifier
                OPTIONAL,

            user-data   -- MAY BE present on user-initiated abort
                UserData-PDU
                OPTIONAL,

            reason[1]   -- ALWAYS present on provider-initiated abort
                IMPLICIT Abort-reason
                OPTIONAL
        }

Abort-reason ::=
        INTEGER {
            unspecified(0),
            unrecognized-ppdu(1),
            unexpected-ppdu(2),
            unrecognized-ppdu-parameter(4),
            invalid-ppdu-parameter(5),
            reference-mismatch(9)
        }


-- data PDU

UserData-PDU ::=
    [5]                             -- this is the ASN.1 object
```

```
        ANY                             -- if it is a top-level PDU, it
                                        -- is in PCI #1, otherwise PCI #3


-- data PDU for the udp-based service

CL-UserData-PDU ::=
    [6]
        IMPLICIT SEQUENCE {
            reference
                SessionConnectionIdentifier,

            user-data[0]                        -- this is the ASN.1 object
                ANY                             -- it is always in PCI #1
        }

END
```

Appendix B:

Example of Serialization

Consider the following call to ROSE:

```
        RO-INVOKE (operation number      = 5
                   operation class       = synchronous
                   argument              = NONE
                   invocation identifier = 1
                   linked invocation id. = NONE
                   priority              = 0)
              .REQUEST
```

Ultimately, ROSE will use the P-DATA service:

```
        P-DATA (user data = {
                              1,        -- this is the PCI
                              {         -- this is the ASN.1 object
                                 invokeID 1,
                                 operation-value 5,
                                 argument {}
                              }
                            })
              .REQUEST
```

The presentation provider will construct a UserData PDU and send this via the transport connection:

```
[5] {
      {
        1,
        5,
        {}
      }
    }
```

Applying the basic encoding rules for ASN.1, we have an stream of 12 octets.

```
a5  0a                                          [5]
tag len

a0  08                                    [0]
tag len
02  01  01          invokeID 1
tag len value

02  01  05          operation-value 5
tag len value

30  00                      : argument NULL
tag len
```

Of course, in actual use, the argument would not be NONE and this could be expected to dominate the size of the UserData PDU. However, it is worth nothing that the overhead of the encoding mechanism used is on the order of 10 octets, hardly a staggering amount!

Appendix C:

Determination of Network Called Address

As described in Section 10, when the P-CONNECT.REQUEST primitive is issued the presentation provider must determine which of the network addresses present in the called presentation address parameter to use for the presentation connection. The first step in this determination is to examine the quality of service parameter and consider only those network addresses which support the corresponding transport service. In practice, it is likely that each network address will support exactly the same transport services, so using quality of service as a discriminant will either permit all or none or the network addresses present to be selected. This appendix describes a local policy which might be employed when deciding which network address to use.

The policy distinguishes between "underlying failures" and

"connection establishment failures". An "underlying failure" occurs when, using the desired transport service, the initiating presentation provider is unable to contact the responding presentation provider. For the tcp-based service, this means that a TCP connection could not be established for some reason. For the udp-based service, it means that a response was not received before final time-out. In contrast, a "connection establishment failure" occurs when the responding presentation provider can be contacted, but the presentation connection is rejected by either the presentation provider or the correspondent presentation user.

The policy is simple: starting with the first network address present, attempt the connection procedure. If the procedure fails due to an "underlying failure", then the next network address in the list is tried. This process is repeated until either an underlying connection is established or all network addresses are exhausted. If, however, a "connection establishment failure" occurs, then the presentation provider immediately indicates this failure to the presentation user and no further network addresses are considered.

Note that this is only one conformant policy of many. For example, the presentation provider may wish to order network addresses based on the "intensity" associated with the members present in the set of transport services for each network address.

Author's Address:

Marshall Rose
The Wollongong Group
1129 San Antonio Road
Palo Alto, CA 94303

Phone: (415) 962-7100

EMail: mrose@TWG.COM